# SPEEDY 1050




# ANWENDER-HANDBUCH

# S P E E D Y   1 0 5 0

## E i n   A n w e n d e r   H a n d b u c h

## u n d   R O M   L i s t i n g


Version 1.0 vom 16.10.86

Autoren:
Peter Bee, Erwin Reuß und Reinhard Wilde

SPEEDY 1050 ANWENDER HANDBUCH


Druck: DER DRUCKER, BOCHUM
Printed in Germany

INHALTSVERZEICHNIS:

| LABEL | SEITE |
|---|---|

DIE BEFEHLE DER SPEEDY 1050 34

---

VORWORT

Wenn Sie dieses Handbuch gekauft haben, sind Sie wahrscheinlich schon Besitzer einer SPEEDY 1050, und kennen die Vorteile, die Ihnen dieser Hardwarezusatz für Ihr Laufwerk bietet, und wollen jetzt etwas mehr über die Möglichkeit der Programmierung der SPEEDY 1050 erfahren.

Genau das wollen wir Ihnen mit diesem ANWENDER HANDBUCH ermöglichen: Die Programmierung Ihrer SPEEDY 1050.

Und genau hierfür haben wir dieses Handbuch geschrieben. Sie sollen die Möglichkeit bekommen, die speziellen Fähigkeiten der SPEEDY 1050 für Ihre eigenen Programme zu nutzen. So können Sie zum Beispiel sehr leicht die SUPER SPEED Routine der SPEEDY 1050 in Ihre Programme einbauen, oder die Fähigkeit der SPEEDY, eine Diskette zu formatieren, ohne daß Sie dazu den Computer oder ein DOS benutzen müssen. Oder Sie könnnen sich selber ein Kopierprogramm schreiben, mit dem Sie Ihre Originaldisketten kopieren können.

Sie werden dazu in diesem ANWENDER HANDBUCH die genaue Dokumentation des ROM-Listings, sowie eine ausführliche Dokumentation der Einsprungadressen und natürlich Demoprogramme finden. Wir glauben, das wir dieses ANWENDER HANDBUCH leicht verständlich geschrieben haben.

---

Wir wenden uns mit diesem ANWENDER HANDBUCH an die Programmierer, die wissen, wie Sie einen Floppy-Controller vom Typ 2797 oder 2793 programmieren müssen. Es würde zu weit führen, die Programmierung dieses Floppy-Controllers in diesem Handbuch zu erklären. Wir möchten in diesem Zusammenhang auch auf die ausgezeichnete Dokumentation des Herstellers dieses Bausteins hinweisen, die Sie in jedem guten Zubehörhandel bekommen können.

Hier also noch einmal: Dieses ANWENDER HANDBUCH ist nicht gedacht für den Anfänger in der Maschinensprach Programmierung! Es ist speziell geschrieben worden für den fortgeschrittenen Programmierer.

Wichtig für einen Programmierer ist auch noch der folgende Hinweis: Sie brauchen an uns keine weitere Lizenzgebühr zu zahlen, wenn Sie Software für die SPEEDY 1050 schreiben, oder Routinen aus der SPEEDY 1050 innerhalb Ihrer Software für die SPEEDY 1050 benutzen. Sie können also Ihre Programme Kommerziell vermarkten. Wir möchten Sie nur bitten, uns eine Kopie Ihres Programmes zu kommen zu lassen.

Und nun viel Spaß bei der Programmierung Ihrer SPEEDY 1050.

Ihr COMPY-SHOP TEAM

---

DER AUFBAU DER SPEEDY 1050 PLATINE

Es gibt zwei verschiedene Versionen der SPEEDY 1050. Die Erste ist die Grundausführung. Die zweite Version der SPEEDY 1050 ist eine Erweiterte Ausführung mit Trackanzeige und einem akustischen Fehlermelder, dem Summer. Technisch, und auf die Laufwerksleistungen bezogen, sind beide Versionen identisch.

DER AUFBAU DER GRUNDVERSION

Die Grundversion besteht aus der Platine, einem 8K RAM-IC, einem 8K EProm mit dem Betriebssystem, dem Mikroprozessor 65C02 und diversen Kodier-IC's.

DER AUFBAU DER ERWEITERTEN VERSION

Zusätzlich zu den Bauteilen der Grundversion kommen bei der erweiterten Version noch die Bauteile für den Summer und die Trackanzeige hinzu.

Die Grundversion lässt sich leicht durch einen entsprechenden Bausatz mit Trackanzeige und Summer nachrüsten.

---

DIE FUNKTIONSWEISE DER SPEEDY 1050

Ein normales ATARI 1050 Laufwerk besitzt einen RAM Buffer von 256 Bytes Größe. Diesen RAM-Bereich müssen sich Datenspeicher und Mikroprozessor teilen. Da für einen Sektor ja bereits 128 Bytes gebraucht werden, hat der Mikroprozessor nicht mehr viel Platz zum arbeiten. Aus diesem Grund kann bei einem ATARI 1050 Laufwerk pro Umdrehung der Diskette jeweils nur ein Sektor eingelesen und zum Computer gesendet werden. In einem Track liegen, bei SINGLE DENSITY, 18 Sektoren. Die Diskette rotiert mit ca. 5 Umdrehungen pro Sekunde (288 Umdrehungen/Minute).

Das ergibt, für einen kompletten Track, eine Ladezeit von ca. 3,6 Sekunden. Das ist die Zeit, die das ATARI 1050 Laufwerk benötigt, um einen kompletten Track einzulesen.

Die SPEEDY 1050 besitzt einen 8K-Byte großen RAM Speicher. Dieser arbeitet als DATENSPOOLER. Pro Umdrehung der Diskette kann nun ein kompletter Track in den RAM-Buffer eingelesen werden. Das ergibt eine Ladezeit für einen Track von 0,2 Sekunden. Die Geschwindigkeit ist also um den Faktor 18 erhöht worden. Diese Geschwindigkeit ist die normale Arbeitsgeschwindigkeit eines SPEEDY 1050 Laufwerkes.

Durch die Zwischenspeicherung der Daten kann nun auch die Übertragungsgeschwindigkeit zum Computer erhöht werden. Durch diesen grossen RAM Buffer können nun auch mehr als 128 Bytes in einem Sektor geschrieben werden. Es wird also echte DOUBLE DENSITY, 256 Bytes per Sektor, möglich. Auch bei dieser Speicherdichte, von nun 176K Byte auf einer Diskettenseite, arbeitet das SPEEDY 1050 Laufwerk mit der hohen Geschwindigkeit.

---

Auf der SPEEDY 1050 Platine befindet sich neben dem RAM aber auch noch ein neuer Mikroprozessor. Dieser Mikroprozessor ist der 65C02. Gegenüber dem 6507 einer normalen ATARI 1050 bietet der 65C02 zwei grosse Vorteile.

Erstens kann der 65C02 bis zu 64K adressieren, der 6507 nur 8K-Bytes, und zweitens besitzt der 65C02 einen erweiterten Befehlssatz mit zusätzlichen, sehr nützlichen Befehlen. Aufgrund dieser zusätzlichen Befehle konnte das Betriebssystem der SPEEDY 1050 kurz gehalten werden, und die Geschwindigkeit der Programmausführung wird durch die geschickte Ausnutzung des erweiterten Befehlssatzes gesteigert.

---

DIE DATENÜBERTRAGUNG ZUM COMPUTER

Bei einem normalen ATARI 1050 Laufwerk wird, wie wir schon erwähnt haben, pro Diskettenumdrehung ein einzelner Sektor eingelesen und dann sofort an den Computer weitergegeben.

Diese Methode ist sehr Zeitraubend. Bei einem SPEEDY 1050 Laufwerk wird, bei einer Diskettenumdrehung, ein kompletter Track in den RAM Buffer eingelesen. Dadurch hat der Computer jederzeit Zugriff auf alle Sektoren, die sich in diesem Track befinden. Durch diese Zwischenspeicherung der Daten ist, im Normalmodus, eine Übertragung der Daten ohne Pause (bedingt durch die Ladezeit zwischen den Sektoren) möglich.

Das bedeutet, daß im Normalmodus die Lesegeschwindigkeit im Laufwerk mit maximaler Geschwindigkeit läuft, die Datenübertragung zum Computer aber mit normaler Geschwindigkeit geschieht. Aber aufgrund der wegfallenden Pausen verkürzt sich die Ladezeit um ca. 50%.

Beim aktivieren der speziellen SPEEDY 1050 Geschwindigkeit, der SUPER SPEED, wird die Datenübertragung vom Laufwerk zum Computer auf das Maximum gesetzt. Bei einem normalen ATARI 1050 Laufwerk geschieht die Datenübertragung zum Laufwerk mit 19.200 Baud. Beim aktivieren der SUPER SPEED erhöht sich diese Baudrate auf das 4 fache.

---

PROGRAMMIEREN DER SPEEDY 1050

Aufgrund der besonderen Fähigkeiten der SPEEDY 1050 haben Sie nun auch die Möglichkeit das Laufwerk individuell zu programmieren.

Die Möglichkeiten, die sich Ihnen damit eröffnen, sind fast unerschöpflich. So können Sie zum Beispiel Diskettenformate nach Ihrem eigenen Bedarf erstellen, einen eigenen Kopierschutz erzeugen oder einen fremden kopieren.

Auch die SUPER SPEED können Sie sehr leicht für Ihre eigenen Programme nutzen, wie Sie anhand eines Demoprogrammes sehen werden.

SPEICHERAUFTEILUNG SPEEDY 1050

Nachfolgend finden Sie ein Blockschaltbild mit der genauen Speicherbelegung der SPEEDY 1050.

| | |
|---|---|
| $FFFF | |
| | R O M - 8 K B Y T E |
| $E000 | |
| $DFFF | |
| | U N B E N U T Z T |
| $A000 | |
| $9FFF | |
| | R A M - 8 K B Y T E |
| $8000 | |
| $7FFF | |
| | U N B E N U T Z T |
| $0404 | |
| $0403 | |
| | C O N T R O L L E R 2 7 9 3 / 9 7 |
| $0400 | |
| $03FF | |
| | U N B E N U T Z T |
| $0300 | |
| $02FF | |
| | P O R T (I/O + T I M E R) |
| $0280 | |
| $027F | |
| | U N B E N U T Z T |
| $0200 | |
| $01FF | |
| | S T A C K |
| $0100 | |
| $00FF | |
| | Z E R O P A G E |
| $0000 | |

---

ERKLÄRUNG ZUR SPEICHERBELEGUNG

$E000 - $FFFF - BETRIEBSSYSTEM

Hier liegt das Betriebssystem Ihrer SPEEDY 1050. Änderungen können Sie hier nicht vornehmen. Ein genaues Listing dieses Betriebssystemes finden Sie im Anhang dieses Handbuches.

$8000 - $9FFF - ARBEITSSPEICHER

Der 8K Ram Block ist in 5 Bereiche unterteilt:

$9F80 - $9FFF - Hier liegen Einsprung und Rücksprungvektoren für die Bereitschafteroutine des Betriebssystems. Außerdem können Sie hier Erweiterungen der RESET Routine vornehmen.

$9F00 - $9F7F - Die normale und erweiterte Kommandotabelle und die entsprechenden Einsprünge sind hier zu finden. Über das Kommando $41 können Sie diese Tabelle beliebig verändern. Diesen Befehl (und andere) haben wir Ihnen bereits im SPEEDY 1050 Handbuch erklärt, Sie werden Sie aber auch noch einmal, etwas später, in diesem ANWENDER HANDBUCH finden.

$9E00 - $9EFF - Der EXTENDED BUFFER dient zur Zwischenspeicherung von Sektordaten bei FAST WRITE oder beim SLOW MODE diverser Laufwerksfunktionen.

$8C00 - $9DFF -In diesem Bereich liegt der Trackbuffer. Hier werden bei FAST WRITE oder FAST READ erst alle Sektordaten eines Tracks zwischengespeichert. Schalten Sie die SPEEDY 1050 zum Beispiel mit Hilfe der Menu-Diskette (Menupunkt SLOW MODE CONTROL) für READ SECTOR und WRITE SECTOR in den SLOW MODUS, wird dieser Speicherbereich nicht mehr vom Betriebssystem benutzt. So können Sie auch hier eigene Routinen ablegen.

$8000 - $8BFF -Freier Speicherbereich, der dem Programmierer ständig zur Verfügung steht, also wo Sie Ihre eigenen Programme ablegen könnnen!

---

$0400 - $0403 - Hier liegen die Register des Disk-Controllers 2793/97:

$0400 : lesen=Statusregister, Schreiben=Commandregister
$0401 : Lesen + Schreiben=Trackregister
$0402 : Lesen + Schreiben=Sektorregister
$0403 : Lesen + Schreiben=Datenregister

$0280 - $02FF - Hier befinden sich die Register des Port IC's 6532 (RIOT)

Die gebräuchlichsten Register:

$0280 : Port A Datenregister
$0281 : Port A Richtungsregister
$0282 : Port B Datenregister
$0283 : Port B Richtungsregister
$0296 : Timer lesen/schreiben, Timer IRQ abschalten
$029F : Timer mit Teilerverhältnis 1:1K lesen/schreiben, Timer IRQ einschalten

$0000 - $00FF - Die Zeropage und die Page 1 überlagern sich. Das heißt, Speicherstelle $0000 entspricht der Speicherstelle $0100, Speicherstelle $0001 entspricht der Speicherstelle $0101, usw. In der Zeropage stehen dem Benutzer die Speicherstellen $0090 bis $00CF zur freien Verfügung.

---

DIE EINSPRUNGADRESSEN

Nachdem wir Ihnen nun die Speicherbelegung der SPEEDY 1050 erklärt haben, erfahren Sie jetzt etwas über die Einsprungadressen. Wichtig ist dabei eines zu wissen: Wir haben am Ende des ROM--Bereiches ab $FF00 eine Jumptable eingerichtet. Wenn Sie nun eine Funktion innerhalb der SPEEDY ausführen wollen, brauchen Sie nur eine, oder mehrere Adressen dieser Jumptable anzuspringen.

Welche Versionen es in Zukunft auch von der SPEEDY 1050 Software geben wird, diese Jumptable wird immer an der gleichen Stelle und in der gleichen Reihenfolge erhalten bleiben. Somit ist gewährleistet, das alle Software, die jetzt für die SPEEDY 1050 geschrieben wird, auch auf den zukünftigen Versionen laufen wird. (Wenn die internen Routinen über diese Tabelle angesprungen werden!)

Hier die Beschreibung der einzelnen Jump Vectoren:

---

RESET - $FF00 - DRIVE KALTSTART

Port A und Port B (6532) werden initialisiert, der komplette RAM Bereich gelöscht und der Disk-Controller getestet. Sollte ein Fehler beim Testen auftreten, erfolgt ein Sprung nach "SYSERO", wo 2 mal "BELL" ausgegeben wird, und anschliessend folgt ein Sprung nach "RESET2".

RESET2 - $FF03 - DRIVE WARMSTART

Die System-Variablen werden neu gesetzt, das Laufwerk in den "SINGLE-DENSITY-MODUS" gebracht, der SchreibLesekopf auf Track 0 justiert und jede Controller Tätigkeit gestopt. Zum zurücksetzen des Laufwerkes sollte diese Routine angesprungen werden, da bei "$FF00 - RESET" alle im RAM befindlichen Daten und Programme gelöscht werden.

BEREIT - $FF06 - BEREITSCHAFTSROUTINE

Zuerst wird der "SLOW" Schalter abgefragt und das Laufwerk gegebenenfalls in den "SLOW" Modus geschaltet. Sollten sich noch zu schreibende Sektoren im RAM befinden, wird direkt nach "TSTCO" verzweigt, damit, falls die Laufwerksklappe geöffnet wird, der Motor nicht ausgeschaltet wird. Wurde nicht verzweigt, wird die Laufwerksklappe mit der letzten Stellung verglichen. Sollte die Klappe geöffnet worden sein, so werden der Antriebsmotor und der Steppermotor ausgeschaltet und der Controller-Status nach "CONST" kopiert. Ist die Klappe geschlossen worden, wird die DENSITY neu festgestellt, das Laufwerk entsprechend eingestellt und die Sektor-Folge neu gelesen. In der "TESTCO" Routine wird die "COMMAND" Leitung vom Computer abgefragt. Sollte diese "gesetzt" werden, erfolgt ein Sprung nach "RDINF", wo das Kommando vom Computer eingelesen wird. Ist die "COMMAND"-Leitung nicht gesetzt wird der Motor-Timer heraufgezählt. Sollte dieser den Wert $9800 (16 Bit) erreichen, wird "TSTWR" aufgerufen um alle Sektoren zu schreiben, die sich noch im RAM befinden.

MOTON - $FF09 - MOTOR ZWINGEND EINSCHALTEN

Der Antriebsmotor wird ohne Rücksicht auf die Stellung der Laufwerksklappe eingeschaltet.

---

TSTMON - $FF0C - MOTOR EINSCHALTEN, WENN DIE LAUFWERKSKLAPPE GESCHLOSSEN IST

Erst wird die Laufwerksklappe abgefragt. Ist diese geschlossen, wird der Antriebsmotor eingeschaltet und eine Zeitverzögerung durchgeführt, um den Motor auf Touren kommen zu lassen.

MOTOFF - $FF0F - MOTOR AUSSCHALTEN

Der Antriebsmotor wird ausgeschaltet und das entsprechende Bit in "DRSTAT" gesetzt.

SDELAY - $FF12 - MOTOR TIMER EINSTELLEN

Diese Routine wird nach einer Kommandoausführung abgearbeitet. Zuerst wird die Zeit gestellt, wie lange der Motor nach einer Kommandoausführung noch eingeschaltet bleiben soll, dann der Kommando-Buffer gelöscht. Danach erfolgt ein Rücksprung in die Motor-Timer-Routine innerhalb der Bereitschafts-Routine.

---

SDRDDP - $FF15 - DRIVE DENSITY EINSTELLEN UND ANZEIGEN

Zuerst wird nach "DENDSP" gesprungen um die aktuelle DENSITY auf dem Display anzuzeigen. Danach wird, je nach Wert in "FORKEN" der Drive-Status, die Anzahl der Sektoren pro Track und die Anzahl der Bytes per Sektor gesetzt.

XWAIT - $FF18 - WARTESCHLEIFE KURZ

Der Wert im X-Register gibt die Anzahl der Schleifendurchläufe an. Ein Schleifendurchlauf entspricht ca. 100 Taktzyklen (0.1 msec/100müsec).

X2WAIT - $FF1B - LANGE WARTESCHLEIFE

Der Wert im X-Register gibt die Anzahl der Schleifendurchläufe an. Ein Schleifendurchlauf entspricht ca. 100.000 Taktzyklen (0.1 sec/100msec).

TRACKO - $FF1E - TRACK 0 POSITIONIEREN

Der Disk-Controller wird gestoppt und der Schreib/Lesekopf so lange zurückgezogen, bis der Track-0-Sensor anspricht. Danach wird eine Zeitverzögerung zum ausschwingen der Kopfmechanik durch geführt.

---

TRADJA - $FF21 - TRACK ANZEIGEN UND SCHREIB/LESEKOPF POSITIONIEREN

Zuerst wird die Tracknummer angezeigt und der Controller gestoppt. Ist die Klappe geschlossen wird der Motor eingeschaltet und die Anzahl der Doppel-Steps errechnet, die durchgeführt werden müssen, um die gewünschte Kopfposition zu erreichen. Sollte der Trackwechsel über 40 Tracks gehen, wird 2*"BELL" ausgegeben und der Kopf auf Track 0 positioniert.

Nach erfolgreicher Schreib/Lesekopf Positionierung wird die Tracknummer in das Track-Register des Controllers kopiert und der Schreib/Lesekopf Mechanik Zeit zum ausschwingen gegeben.

TRADJ - $FF24 - SCHREIB-LESE-KOPF POSITIONIEREN

Entspricht der $FF21 Routine, mit dem Unterschied, das die Tracknummer nur dann angezeigt wird, wenn ein Trackwechsel statt gefunden hat.

TRVR - $FF27 - 1 STEP VORWÄRTS ODER RÜCKWÄRTS AUSFÜHREN

Im Y-Register ist die Kennung für Step vorwärts oder rückwärts (plus oder minus). Die Bitposition des Steppermotors wird entsprechend heraufoder herabgezählt und das neue Bitmuster in Port B des Portbausteins geschrieben.

Anmerkung:
Für 1 Trackwechsel müssen immer 2 Steps erfolgen. Das heißt aber nicht, daß ohne bedenken 80 Tracks geschrieben oder gelesen werden können. Die beiden Steps sind verschieden lang!

CONRES - $FF2A - DISK CONTROLLER STOPPEN

Dem Disk-Controller wird der Befehl gegeben, alle laufenden Aktionen zu stoppen. In "WREADY" wird gewartet, bis das der Controller den Befehl als "ausgeführt" meldet.

CONRE2 - $FF2D - $FF2A WIRD ZWEIMAL AUSGEFÜHRT

Beim Einsprung in diese Routine wird die Routine bei $FF2A - DISK CONTROLLER STOPPEN zweimal ausgeführt.

WREADY - $FF30 - AUF CONTROLER "IN USE" FLAG=0 WARTEN

Bei dieser Routine wird darauf gewartet, daß der Controller meldet, daß der letzte Befehl abgeschlossen wurde.

RD128 - $FF33 - 128 BYTES VOM COMPUTER NACH "EXBUF" HOLEN

Der Buffer wird auf "EXBUF" (Extended Buffer) und die IO-Länge auf 128 Bytes gesetzt.

RD256 - $FF36 - 256 BYTES VOM COMPUTER NACH "EXBUF" HOLEN

Wie bei der vorhergehenden Routine wird der Buffer auf "EXBUF", aber die IO-Länge auf 256 Bytes gesetzt.

---

RDBTS - $FF39 - ANZAHL BYTES IN DEN BUFFER HOLEN (X/Y REGISTER)

Im Accu stehen die Anzahl der Bytes, im X- und Y-Register die Low- und High-Adresse des Buffers, in die die Daten vom Computer abgelegt werden sollen. Der Timer wird gesetzt, um zu verhindern, das der Prozessor hängen bleiben kann. Es wird jeweils ein Byte über einen indirekten Jump-Vector vom Computer geholt (ausser bei HIGH SPEED) und in dem Buffer abgelegt. Anschließend wird die Checksumme heraufgezählt und geprüft, ob alle Bytes und Datenblocks geholt wurden. Die Checksumme wird verglichen und die IO-Länge neu gesetzt.

RDSFOL - $FF3C - NACH VERZÖGERUNG SEKTORFOLGE VOM AKTUELLEM TRACK LESEN

Die Verzögerungsschleife am Anfang der Routine verhindert, daß versucht wird, die HEADER schon zu lesen, wenn die Laufwerks-Klappe noch nicht vollständig geschlossen ist. Danach wird die Zeit gestellt, die der Controller zur Verfügung hat, um alle HEADER zu lesen. Ist ein HEADER eingelesen, wird die Track- und Sektor- Nummer auf Gültigkeit geprüft. Befindet sich die Sektornummer des gelesenen HEADERs bereits in der Sektorliste (doppelter Sektor), so wird die Sektor-Folge nicht mehr weiter gelesen und das Laufwerk in den Slow-Modus geschaltet. Stimmt die Anzahl der gelesenen Sektor-HEADER nicht mit der vorgegebenen (18 oder 26 Sektoren) überein, wird das Laufwerk ebenfalls in den Slow-Modus geschaltet.

---

RDTRA - $FF42 - AKTUELLEN TRACK INS RAM EINLESEN

Die Track-Slow-Kennung wird gesetzt. Über RDHDSP wird der Kopf positioniert und der nächste HEADER eingelesen. Anschließend wird die Position der Sektor-Nummer des HEADERs in der Sektorliste gesucht. Ab der nächsten Position werden die Sektoren dann in der richtigen Reihenfolge eingelesen. Die Statuswerte der Sektoren werden in die Statusliste eingetragen. Statuswerte, ausser $08 (CRC-ERROR) und $20 (AM-ERROR) unterbrechen dabei das Einlesen der Sektoren. Nur wenn alle Sektoren des Tracks gelesen wurden, wird die Kennung für Track-Slow zurückgesetzt.

RDTRAV - $FF45 - WIE $FF42, JEDOCH MIT VERIFY UND EINEM RETRY

Hier wird zuerst die Track- und Sektornummern errechnet und anschließend $FF42 (RDTRA) aufgerufen. Dann wird die Statusliste auf $08 (CRC-ERROR) und $20 (AM-ERROR) getestet. Ist einer der beiden Statuswerte eingetragen, wird der entsprechende Sektor nochmals gelesen.

---

TSTWR - $FF48 - NOCH ZU SCHREIBENDE SEKTOREN AUS DEM RAM AUF DIE DISKETTE SCHREIBEN

In "WRKEN" steht die Anzahl der zu schreibenden Sektoren. Ist der Wert=0 wird die Routine sofort verlassen. Sonst wird der Motor eingeschaltet, die Tracknummer für die zu schreibenden Sektoren aus "LWRTRA" nach "TRACK" kopiert und der Schreib/Lesekopf positioniert. Anschließend wird die nächste HEADER eingelesen und die Sektornummer in der Sektorliste gesucht. Ist der Sektor nicht eingetragen (zum Beispiel bei geschützten Disketten, die mit "FAST-WRITE" beschrieben wurden), wird 1 Bell ausgegeben. Ansonsten wird ab der gefundenen Sektorposition die Statusliste auf negative Werte (Kennung für zu schreibende Sektoren) überprüft. Wird ein solcher Wert gefunden, wird der entsprechende Sektor geschrieben und in der Statusliste als "geschrieben" ($40) eingetragen. "WRKEN" wird um eins herunter gezählt und die Position in der Sektorliste heraufgezählt.

Sollte beim schreiben eines Sektors ein Fehler auftreten, in dem der "WRITE PROTECT-SCHALTER" umgeschaltet oder die Laufwerksklappe geöffnet wurde, so wird die Routine "WRERR" aufgerufen, in der dem Anwender auf optischem und akustischem Wege ca. 5 Sekunden Zeit gegeben wird, um die Laufwerksklappe wieder zu schliessen. Alle "WRITE-PROTECTIERTEN" Sektoren werden in der Statusliste als geschrieben ($40) eingetragen.

---

TSTDAT - $FF4B - $FF48 AUFRUFEN UND STATUSLISTE MIT $40 FÜLLEN

Diese Routine muß abgearbeitet werden, wenn das Laufwerk in den "SLOW-MODE" geschaltet wurde, damit alle Sektoren als geschrieben bzw. als nicht gelesen markiert werden.

SD128B - $FF4E - 128 BYTES VOM EXBUF ZUM COMPUTER SENDEN

Die Übertragungslänge wird auf 128 Bytes und der Buffer auf "EXBUF" gesetzt.

SD256B - $FF51 - 256 BYTES VOM EXBUF ZUM COMPUTER SENDEN

Die Übertragungslänge wird auf 256 Bytes und der Buffer auf "EXBUF" gesetzt.

SDBTS - $FF54 - ACCU=ANZAHL DER BYTES AUS BUFFER (X/Y-REGISTER) SENDEN

Es wird jeweils ein Byte aus dem Buffer geladen, die Checksumme heraufgezählt und über die Jump--Tabelle der Senderoutinen gesprungen, um das gelesene Byte zum Computer zu senden. Dieser Vorgang wird in einer Schleife solange wieder holt, bis der gesammte Buffer gesendet wurde. Anschließend wird die Checksumme gesendet.

---

SEND41 - $FF57 - QUITTUNG $41 ZUM COMPUTER SENDEN

Der Accu wird mit dem Wert $41 (A) geladen und nach einer Verzögerung (damit die Quittung an den Computer nicht zu schnell kommt) über die Jump-Tabelle der Senderoutinen gesprungen, um die Quittung in der richtigen Übertragungsgeschwindigkeit zu senden.

SEND43 - $FF5A - QUITTUNG $43 ZUM COMPUTER SENDEN

Das gleiche wie bei "SEND41"-$FF57, jedoch mit dem Quittungsbyte $43 (C).

SEND45 - $FF5D - QUITTUNG $45 ZUM COMPUTER SENDEN

Das gleiche wie bei "SEND41"-$FF57, jedoch mit dem Quittungsbyte $45 (E).

SEND4E - $FF60 - QUITTUNG $4E ZUM COMPUTER SENDEN

Das gleiche wie bei "SEND41"-$FF57, jedoch mit dem Quittungsbyte $4E (N).

RDSECT - $FF63 - AKTUELLEN SEKTOR VON DER DISKETTE IN DEN VORBEZEICHNETEN RAM BEREICH EINLESEN

Sektornummer in das Sektor-Register kopieren. "READ-SEKTOR"- Befehl an den Controller geben. "TIME-OUT"-Zeit setzen. Nun werden die Daten Byte für Byte vom Controller übernommen und in den bezeichneten Buffer (indirekt "IND") abgelegt. Sind alle Daten gelesen wird auf den Controller gewartet, bis dieser seine Arbeit eingestellt hat und der Status des gelesenen Sektors in das Status-Register des Controllers übernommen. Sollte ein "TIME-OUT" aufgetreten sein, wird noch ein Leseversuch gestartet.

RDSEC1 - $FF66 - BEZEICHNETEN SEKTOR IN BEZEICHNETEN RAM EINLESEN

Die selbe Routine wie RDSECT- $FF63. Nur muß die Sektornummer bereits in das Sektor-Register des Controllers geschrieben sein.

---

WRSECT - $FF69 - AKTUELLEN SEKTOR VON VORBEZEICHNETER RAMADRESSE AUF DIE DISKETTE SCHREIBEN

Sektornummer in das Sektor-Register kopieren. "WRITESECTOR"- Befehl an den Controller geben. "TIME-OUT"-Zeit setzen. Nun werden die Daten Byte für Byte an den Controller übergeben.

Sind alle Daten übergeben, wird auf den Controller gewartet, bis dieser seine Arbeit eingestellt hat, und der Write-Status vom Controller übernommen wird. Sollte ein "TIME-OUT" aufgetreten sein, wird überprüft, ob der Controller noch arbeitet. Wenn ja, wird noch ein Schreibversuch gestartet.

WRSEC1 - $FF6C - BEZEICHNETEN SEKTOR VON BEZEICHNETER RAMADRESSE AUF DISKETTE SCHREIBEN

Die selbe Routine wie WRSECT- $FF69. Nur muß die Sektornummer bereits in das Sektor- Register des Controllers geschrieben sein.

TSTWRP - $FF6F - "WRITE PROTECT" UND LAUFWERKSKLAPPE TESTEN

Es wird "CONRES" aufgerufen, wo der Disk-Controller seine augenblickliche Arbeit unterbricht und der Controller Status gelesen wird. Anschließend werden Bit 6 (WRITE PROTECT) und Bit 7 (LAUFWERKS-KLAPPE) ausmaskiert. Ist eines der beiden Bits gesetzt, können keine Daten geschrieben werden!

---

VERSEC - $FF72 - AKTUELLEN SEKTOR MIT ANGEGEBENEN RAM VERGLEICHEN

Sektornummer in das Sektor-Register kopieren. "READ-SEKTOR"-Befehl an den Controller geben. "TIME-OUT"- Zeit setzen.

Nun werden die Daten Byte für Byte vom Controller übernommen und mit der bezeichneten Adresse (indirekt "IND") verglichen. Ist ein Wert ungleich, wird das lesen unterbrochen, der Controller gestoppt, die Kennung für "DATEN UNGLEICH" (ACCU <>0) gesetzt und CARRY für "KEIN LESE FEHLER AUFGETRETEN" gelöscht.

Stimmen alle gelesenen Daten mit dem angegebenen Buffer überein, wird die Kennung für "DATEN GLEICH" (ACCU=0) und "KEIN LESEFEHLER" (CARRY=0) gesetzt. Tritt während des Vergleiches ein "TIME OUT" auf, wird geprüft, ob der Controller noch arbeitet. Wenn ja, wird der Vergleich der Daten fortgesetzt. Ansonsten wird "CARRY" gesetzt (KENNUNG FÜR LESEFEHLER).

VERSE1 - $FF75 - BEZEICHNETEN SEKTOR MIT ANGEGEBENEM RAM VERGLEICHEN

Die selbe Routine wie VERSEC - $FF72. Nur muß die Sektornummer in das Sektor-Register des Controllers geschrieben sein.

STELL - $FF78 - COM-STATUS AUF ERROR UND 2 RETRY'S SETZEN

"RETRY" wird auf zwei Versuche und "COMST" vorsorglich auf "COMMAND-ERROR" gesetzt.

---

QUITT - $FF7B - QUITTUNG "C" ODER "E" ZUM COMPUTER SENDEN

Den Controller Status übernehmen. Wenn "CONST" auf "COMMAND ERROR" steht, wird die Kennung für "FEHLER BEI LETZTER LAUFWERKS-OPERATION" in "DRSTAT" gesetzt. Wenn Bit 7 und Bit 0 in "DSPCTR" gesetzt sind, wird der Controller Status auf dem Display angezeigt und ein "BELL" ausgegeben, und die Quittung $45 ("E") zum Computer gesendet. Ist "COMMAND STATUS" o.k. wird die Kennung für "LAUFWERKS-OPERATION-IN-ORDNUNG" gesetzt und die Quittung $43 ("C") zum Computer gesendet.

RDHEAD - $FF7E - DIE NÄCHSTEN HEADER DATEN LESEN

"TIME OUT"-Zeit setzen. "READ HEADER"-Befehl an den Controller geben. Nun werden die 6 Bytes des nächsten auffindbaren "HEADER'S" in einer Schleife eingelesen und ab der Adresse $7A abgelegt. In "WREADY" wird darauf gewartet, daß der Controller seine Arbeit einstellt. CARRY als Lesefehler-Flag wird zurückgesetzt.

RDHD1 - $FF81 - WIE RDHEAD, ABER TIMER NICHT SETZEN

Die gleiche wie $FF7E. Nur muß "TIME OUT" bereits gesetzt sein.

CALCTS - $FF87 - TRACK- UND SEKTORNUMMER ERRECHNEN

Sektornummer LOW und HIGH werden zum "IND"-Pointer kopiert (für RAM/ROM Adressen) und auf Nummer=0 oder Nummer größer als $7FFF geprüft. Ist das der Fall, wird in "DUMKEN" noch der SLOW-Status getestet. Andernfalls wird die Sektornummer in IND/IND+1 solange um die Anzahl der Sektoren pro Track herabgezählt, bis diese die Nummer Null unterschreitet. Als Ergebnis hat man die gewünschte Track- und Sektornummer. Die Tracknummer wird noch mit dem Wert 40 verglichen. Das CARRY-Flag wird durch den Vergleich entsprechend gesetzt. Nach der Rückkehr aus dieser Routine stehen die Prozessor Status-Flags wie folgt:

C=1 SEKTORNUMMER UNZULÄSSIG
N=1 RAM/ROM ADRESSE
Z=1 ZERO-PAGE ADRESSE 0

SETBUF - $FF8A - BUFFER NACH AKTUELLEM SEKTOR SETZEN

Je nach Sektorlänge wird die Sektornummer durch zwei geteilt und zur Anfangsadresse des Datenbuffers hinzu addiert. Die Bufferadresse befindet sich dann in IND/ IND+1.

SETBUF2 - $FF8D - BUFFER NACH SEKTORNUMMER IM ACCU SETZEN

Entspricht der Routine "SETBUF" - $FF8A, jedoch muß die Sektornummer (1...26) bereits im ACCU stehen.

SEXBUF - $FF90 - ADRESSE DES EXTENDED BUFFERS SETZEN

Die Adresse von "EXBUF"wird nach IND/IND+1 geladen.

SETRWL - $FF93 - ANZAHL DER BYTES FÜR ZU ÜBERTRAGENDEN DATENBLOCK SETZEN

Bei Sektornummern von 1 bis 3 wird die Übertragungslänge auf 128 Bytes, ansonsten auf den Wert von "SECLEN" gesetzt. Anzahl der Datenblocks wird auf 1 gesetzt.

COPSLT - $FF96 - SEKTORLISTE FÜR AKTUELLE DENSITY IN ZERO PAGE KOPIEREN

In "SDRDDP" werden die Werte für die aktuelle DENSITY richtig gesetzt und die DENSITY auf dem Display angezeigt. Anschließend wird je nach Wert in "FORKEN" (DENSITY-Kennung) die entsprechende Sektorliste für SINGLE- oder DOUBLE-DENSITY nach "SECLST" ($20) kopiert.

---

BELL1 - $FF99 - 1 MAL BELL ÜBER DEN SUMMER AUSGEBEN

Der Summer wird mittels einer Verzögerungsschleife mit einer bestimmten Frequenz angesteuert.

CLRDSP - $FF9C - DISPLAY ABSCHALTEN

In die Display Speicherstellen $4000, $4001 und $4002 werden Nullen geschrieben. Die Speicherstellen sollten nur im Schreibzugriff (z.b. STA $4000) angesprochen werden, da sonst auf dem Display unkotrolierbare Zeichen erscheinen.

TRAANZ - $FF9F - AKTUELLE TRACKNUMMER ANZEIGEN

Der ACCU wird mit dem Wert von "TRACK" geladen, und es wird, je nach Wert in "DSPCTR" zur Dezimal- oder Hexadezimal Ausgaberoutine gesprungen.

---

DEZOUT - $FFA2 - WERT IM ACCU WIRD IN DEZIMALER FORM ANGEZEIGT

Der Wert im ACCU wird in einer Schleife um 10 heruntergezählt, bis er den Wert 0 unterschreitet. Der Schleifenzähler entspricht dann dem Wert für das 10'ner Stellen- Display, und die Restsumme dem Wert für das 1'ner Stellen- Display. Die Werte für die richtige Segmentsteuerung wird einer Konstantentabelle (SEGTBL) entnommen.

HEXOUT - $FFA8 - WERT IM ACCU WIRD IN HEXADEZIMALER FORM ANGEZEIGT

Zuerst werden die unteren vier Bits des Wertes im ACCU ausmaskiert, die dem Wert für das rechte Display entsprechen, dann die oberen vier Bits.

DENDSP - $FFA8 - AKTUELLE DENSITY AUF DEM DISPLAY ANZEIGEN

Je nach Wert in "FORKEN" (DENSITY Kennung) werden die entsprechenden Segmente des Display's angesteuert.

---

SETTIM - $FFAB - TIMER MIT DEM WERT IM ACCU SETZEN

Timer Interrupt Flag wird gelöscht und der Timer mit dem Wert im ACCU gestartet.

CLRTRA - $FFAE - EINEN TRACK REFORMATIEREN

In "FSTART" wird das "WRITE-TRACK"-Kommando gestartet und der Timer gesetzt. Nun wird der Track mit dem Wert $AA beschrieben, bis der Timer abgelaufen ist. Es wird der Track "gelöscht", auf dem sich der Schreib/Lesekopf befindet.

CLRDSK - $FFB1 - GANZE DISKETTE REFORMATIEREN

Hier werden alle Track's nacheinander, beginnend bei Track 39 (39...0) gelöscht. Der Schreib/Lesekopf wird jeweils positioniert und "CLRTRA" aufgerufen.

RAMTST - $FFB4 - TEST DES LAUFWERK INTERNEN RAM'S

Im ersten Teil wird die Zero Page getestet. Der Wert der Speicherstelle, die getestet wird, wird jeweils um EXBUF+1 zwischengespeichert. Zuerst wird die Zero Page mit dem Wert $55 getestet, daß heißt, der Wert $55 wird in die Speicherstelle geschrieben und wieder gelesen. Ist der Wert gleich geblieben, ist die Speicherstelle in Ordnung. Anschließend wird die Zero Page noch einmal mit dem Wert $AA getestet.

Ist während des Testes kein Fehler festgestellt worden, wird der Speicherbereich von $8000 bis RAMTOP ($A000) auf die gleiche Art getestet, wie die Zero Page. Tritt bei einer Speicherstelle ein Fehler auf, wird die Adresse jener Speicherstelle in $90/$91 abgelegt und der Test abgebrochen.

Ist kein RAM- Fehler festgestellt worden, steht in $90/ $91 die höchste RAM-Adresse. Nach Abschluß der RAM-Testroutine wird die Adresse, die in $90/ $91 steht, zum Computer gesendet.

ROMTST - $FFB7 - ROM TEST

Vorsorglich wird Command-Status auf ERROR gesetzt. In IND/IND+1 wird die Adresse $E000 gesetzt. Anschliessend wird für eine Page die Checksumme errechnet und mit dem Originalwerten in einer Tabelle verglichen. Ist die Checksumme gleich, wird die High-ROM-Adresse in $91 um eins heraufgezählt und die nächste ROM-Page getestet. Insgesamt werden so 32 ROM-Pages ($E000 bis $FFFF) getestet. Stimmen alle Checksummen mit denen der "ROMCHK" Tabelle überein, wird der Command--Status zurück gesetzt und die Quittung ("C") zum Computer gesendet. Ist ein Fehler gefunden worden, wird Command-Status nicht zurückgesetzt und die Quittung ("E") zum Computer gesendet.

SPEEDT - $FFBA - MOTOR SPEED TEST

Vorsorglich Command-ERROR setzen und Schreib/Lesekopf auf Track 0 positionieren. In "FDSEC1" wird Sektor 1 zweimal direkt nacheinander gelesen, und die Zeit über Taktzyklen gemessen. Dann wird in einer Schleife die gemessenen Zeit von der Konstanten $C0E1E4 solange abgezählt, bis der Wert 0 unterschritten wird. Die folgende Nachkomma Stellenrundung wird mit dem Rest der vorhergehenden Rechnung vorgenommen, in dem die gemessene Zeit durch zwei geteilt und von dem Rest der vorhergehenden Rechnung abgezogen wird. Ist das CARRY-Flag dann gesetzt, wird die Nachkommastelle um eins erhöht. Der aus zwei Daten bestehende Speed-Wert wird zum Computer gesendet. Der Hexadezimale Wert $2875 bedeutet dabei 287,5 UPM.

---

DIE BEFEHLE DER SPEEDY 1050

So, nachdem Sie nun mehr über die Einsprungadressen wissen, und bevor wir zum ROM-Listing kommen, hier nun noch einmal alle Befehle der SPEEDY 1050 und Ihre Anwendung.

KOMMANDO ist der Wert, der sich vor Aufruf der SIO - Routine ($E459) in der Speicherstelle $0302 befindet.

AUX1 und AUX2 entsprechen den Werten, die sich in den Speicherstellen $030A und $030B befinden. Bei einigen Befehlen werden AUX1 und AUX2 nicht benutzt und dürfen beliebige Werte annehmen.

Die Befehle sind im übrigen nicht nach den Hexadezimalnummern geordnet, sondern nach Ihrer Funktion!

---

| | |
|---|---|
| KOMMANDO: | $52 |
| FUNKTION: | Sektor lesen |
| AUX1: | Sektornummer oder ROM-Adresse LOW BYTE |
| AUX2: | Sektornummer oder ROM-Adresse HIGH BYTE |
| BESCHREIBUNG: | Es werden je nach DENSITY 128 oder 256 Bytes gesendet. Sektoren 1-3 sind immer 128 Bytes lang. |

| | |
|---|---|
| KOMMANDO: | $50 |
| FUNKTION: | Sektor schreiben ohne Verify |
| AUX1: | Sektornummer oder ROM-Adresse LOW BYTE |
| AUX2: | Sektornummer oder ROM-Adresse HIGH BYTE |
| BESCHREIBUNG: | Das Laufwerk erwartet je nach DENSITY 128 oder 256 Bytes. Sektoren 1-3 sind immer 128 Bytes lang. |

| | |
|---|---|
| KOMMANDO: | $57 |
| FUNKTION: | Sektor schreiben mit Verify |
| AUX1: | Sektornummer oder ROM-Adresse LOW BYTE |
| AUX2: | Sektornummer oder ROM-Adresse HIGH BYTE |
| BESCHREIBUNG: | Das Laufwerk erwartet je nach DENSITY 128 oder 256 Bytes. Die Sektoren 1-3 sind immer 128 Bytes lang. |

---

KOMMANDO: $53
FUNKTION: Laufwerkstatus
AUX1: nicht Benutzt
AUX2: nicht Benutzt
BESCHREIBUNG: Das Laufwerk sendet 4 Bytes, die den Status der letzten Diskettenoperation beinhalten.

Byte 1: Drive Status
- Bit 0 - COMMAND FRAME ERROR
- Bit 1 - CHECKSUM ERROR
- Bit 2 - OPERATION ERROR
- Bit 3 - WRITE PROTECT
- Bit 4 - MOTOR ON
- Bit 5 - DOUBLE DENSITY
- Bit 6 - unbenutzt
- Bit 7 - DUAL DENSITY

Byte 2: Controller Status
- Bit 0 - BUSY
- Bit 1 - DRQ
- Bit 2 - LOST DATA
- Bit 3 - CRC ERROR
- Bit 4 - RECORD NOT FOUND
- Bit 5 - RECORD TYPE
- Bit 6 - WRITE PROTECT
- Bit 7 - NOT READY

Byte 3: Time-Out Wert für Format Disk ($E0)
Byte 4: unbenutzt (immer 0)

---

| | |
|---|---|
| KOMMANDO: | $21 |
| FUNKTION: | Formatiere Diskette (SINGLE/DOUBLE DENSITY) |
| AUX1: | nicht benutzt |
| AUX2: | nicht benutzt |
| BESCHREIBUNG: | Dieses Kommando wird benutzt um Disketten in SINGLE- oder DOUBLE - DENSITY (720 Sektoren) zu formatieren. Das DENSITY-Format wird durch einen vorherigen $4F - Befehl (Laufwerkskonfiguration) eingestellt. Wird das Laufwerk nach dem Einschalten nicht konfiguriert, wird automatisch in SINGLE-DENSITY formatiert. Das Laufwerk sendet nach dem formatieren je nach DENSITY 128 oder 256 Bytes an den Computer. Die ersten zwei Bytes müssen immer $FF sein. |

| | |
|---|---|
| KOMMANDO: | $22 |
| FUNKTION: | Formatiere Diskette (MEDIUM DENSITY) |
| AUX1: | nicht benutzt |
| AUX2: | nicht benutzt |
| BESCHREIBUNG: | Dieses Kommando wird benutzt um Disketten in 1050 DUAL DENSITY (MEDIUM = 1040 Sektoren) zu formatieren. Es werden immer 128 Bytes zum Computer gesendet. Die ersten beiden Bytes müssen immer $FF sein. |

---

KOMMANDO: $20
FUNKTION: Automatisches Formatieren
AUX1: Konfigurationsbyte
AUX2: nicht benutzt
BESCHREIBUNG: Dem Laufwerk wird nur der Befehl zum Formatieren gegeben. Es wird sofort ein 'Complete' zurückgesendet. Mit diesem Befehl können alle drei Formate, abhängig vom Konfigurationsbyte (00=SINGLE, $20= DOUBLE, $80=MEDIUM) generiert werden. Ein Write-Protect wird sofort zurückgemeldet.

Fehler beim formatieren können dem Computer nicht gemeldet werden, da keine Daten nach Befehlsausführung zurückgesendet werden. Der Formatierungsvorgang und eventuelle Formatierungsfehler können auf dem Display verfolgt werden. Abhängig vom Drive/Display - Status Befehl wird nach dem formatieren automatisch die VTOC (Dos 2.5 kompatibel) und 3 Bootsektoren geschrieben.

KOMMANDO: $3F
FUNKTION: SIO - Geschwindigkeitsbyte ermitteln
AUX1: nicht benutzt
AUX2: nicht benutzt
BESCHREIBUNG: Es wird ein Byte zum Computer gesendet, das die HIGH SPEED Übertragungsgeschwindigkeit beinhaltet. Dieses Byte wird für die HIGH SPEED SIO - Routine benötigt und beträgt bei der SPEEDY 1050 normalerweise $09.

---

KOMMANDO: $4E
FUNKTION: Laufwerkskonfiguration auslesen
AUX1: nicht benutzt
AUX2: nicht benutzt
BESCHREIBUNG: Es werden die 12 Bytes der Konfigurationstabelle zum Computer gesendet. Die Bedeutung der einzelnen Bytes sind:

Byte 1 - Anzahl der Tracks (immer 40)
Byte 2 - Step Rate (immer 1)
Byte 3 - Sektoren pro Track HIGH (immer 0)
Byte 4 - Sektoren pro Track LOW (18 oder 26)
Byte 5 - Anzahl der Köpfe (immer 0)
Byte 6 - Aufzeichnungformat (0=FM/4=MFM)
Byte 7 - Bytes pro Sektor HIGH (1=256/0=128)
Byte 8 - Bytes pro Sektor LOW (0=256/128=128)
Byte 9 - Laufwerk aktiv (immer 255)
Byte 10 - unbenutzt (immer 0)
Byte 11 - unbenutzt (immer 0)
Byte 12 - unbenutzt (immer 0)

KOMMANDO: $4F
FUNKTION: Laufwerk konfigurieren
AUX1: nicht benutzt
AUX2: nicht benutzt
BESCHREIBUNG: Dieser Befehl wird benutzt um das Laufwerk für den nähsten Formatierungsbefehl einzustellen. Das Laufwerk erwartet 12 Bytes, die genau der Reihenfolge des vorherigen Befels ($4E) entsprechen müssen.

---

KOMMANDO: $51
FUNKTION: Schreibvorgang beenden
AUX1: nicht benutzt
AUX2: nicht benutzt
BESCHREIBUNG: Nach jedem Schreibbefehl wartet das Laufwerk ca. 2 Sekunden bis die Daten aus dem Trackbuffer auf die Diskette geschrieben werden. Dieses wird durch den Befehl $51 beschleunigt. Alle Daten im Trackbuffer werden unverzüglich auf die Diskette geschrieben und abhängig vom Drive/Display Befehl ($44) wird der Motor nach erfolgtem Schreibvorgang sofort gestoppt.

KOMMANDO: $44
FUNKTION: Drive/Display Einstellung
AUX1: Konfigurationsbyte
AUX2: nicht benutzt
BESCHREIBUNG: Der Wert in AUX1 setzt das Drive/Display Byte im Laufwerk. Dieses Byte kann über keinen Befehl direkt ausgelesen werden, so daß immer alle Bits richtig gesetzt werden müssen.

Die einzelnen Bits beinhalten die folgenden Funktionen:

Bit 0 - BELL bei ERROR zulassen
Bit 1 - unbenutzt
Bit 2 - unbenutzt
Bit 3 - bei Kommando $51 Motor nicht aus schalten
Bit 4 - bei Kommando $20 VTOC+Boot Sektoren nicht schreiben
Bit 5 - Formatieren ohne Verify
Bit 6 - Trackanzeige in Hexadezimal
Bit 7 - ERROR - Anzeige zulassen

---

KOMMANDO: $4B
FUNKTION: Slow/Fast Konfiguration
AUX1: Konfigurationsbyte
AUX2: nicht benutzt
BESCHREIBUNG: Mit dem Wert in AUX1 wird das Drive-Slow-Status Byte des Laufwerkes beeinflußt. Dieses Byte kann über keinen Befehl direkt ausgelesen werden, so daß alle Bits richtig gesetzt werden müssen.

Die einzelnen Bits haben folgende Funktionen:

Bit 0 - Read Sektor slow
Bit 1 - Write Sektor slow
Bit 2 - Kommando $57 Verify ausschalten
Bit 3 - Laufwerk vollständig im 'Slow-Mode'
Bit 4 - unbenutzt
Bit 5 - unbenutzt
Bit 6 - 1 Track slow (nach Trackwechsel 0)
Bit 7 - 1 Diskette slow (nach Disketten-wechsel 0)

KOMMANDO: $4C
FUNKTION: Direkter Sprungbefehl ohne Rückmeldung
AUX1: Sprungadresse LOW BYTE
AUX2: Sprungadresse HIGH BYTE
BESCHREIBUNG: Der Prozessor im Laufwerk wird durch diesen Befehl veranlaßt, direkt zur Speicherstelle zu springen, die sich in AUX1 und AUX2 befindet. Das Laufwerk gibt keine Rückmeldung an den Computer zurück, so daß eine Rückmeldung von dem Programm aus gegeben werden muß, zu dem der Prozessor gesprungen ist.

---

KOMMANDO: $4D
FUNKTION: Direkter Sprungbefehl mit Rückmeldung
AUX1: Sprungadresse LOW BYTE
AUX2: Sprungadresse HIGH BYTE
BESCHREIBUNG: Dieser Befehl gleicht dem Vorhergehenden bis auf den kleinen Unterschied, daß das Laufwerk vor ausführen des Programms eine Rückmeldung an den Computer gibt.

KOMMANDO: $41
FUNKTION: Kommandotabelle verlängern oder verkürzen
AUX1: nicht benutzt
AUX2: nicht benutzt
BESCHREIBUNG: Das Laufwerk erwartet 3 Bytes vom Computer. Das 1. Byte ist das neue Kommando. Das 2. und 3. Byte ist die Startadresse des über das neue Kommando erreichten Programmes in LOW/HIGH-Byte Format. Falls sich der neue Befehl schon in der Kommandotabelle befindet, wird dieser mit der neuen Startadresse versehen. Ist die Startadresse $0000 wird der Befehl aus der Kommandotabelle gelöscht.

---

KOMMANDO: $68
FUNKTION: Länge der SIO - Routine ermitteln
AUX1: nicht benutzt
AUX2: nicht benutzt
BESCHREIBUNG: Mit diesem Befehl wird die Länge der SIO - Routine ermittelt, die mit dem Befehl $69 aus dem Laufwerk in den Computer geladen wird. Das Laufwerk sendet 2 Bytes, die die Länge (LOW/HIGH) beinhalten.

KOMMANDO: $69
FUNKTION: SIO - Routine zum Computer senden
AUX1: Relokator - Adresse LOW BYTE
AUX2: Relokator - Adresse HIGH BYTE
BESCHREIBUNG: Dieser Befehl sendet die HIGH SPEED SIO - Routine mit der vom Befehl $68 ermittelten Länge zum Computer. Diese Routine wird bereits im Laufwerk zur Startadresse hin Relokiert, die sich in AUX1 und AUX2 befindet.

---

KOMMANDO: $60
FUNKTION: Track schreiben
AUX1: Track Anfangssektor oder Anfangsadresse LOW BYTE
AUX2: Track Anfangssektor oder Anfangsadresse HIGH BYTE
BESCHREIBUNG: Die kompletten Daten für einen Track werden mit diesem Befehl auf die Diskette oder in den Trackbuffer geschrieben. Die Anzahl der zu übertragenden Bytes errechnet sich aus der Anzahl der Sektoren mal der Bytes pro Sektor. Wegen des sehr schwierigen Timings funktioniert dieser Befehl nur in normaler Übertragungsgeschwindigkeit.

KOMMANDO: $62
FUNKTION: Track lesen
AUX1: Track Anfangssektor LOW BYTE
AUX2: Track Anfangssektor HIGH BYTE
BESCHREIBUNG: Lesen eines kompletten Tracks mit einem Befehl von der Diskette oder aus dem Trackbuffer. Die Anzahl der zu erwartenden Bytes errechnet sich aus der Anzahl der Sektoren mal der Bytes pro Sektor.

Erlaeuterungen zur Sprungtabelle 'SPEEDY 1050'

| | | | | Unveraenderte Register | ERROR-Flags |
|---|---|---|---|---|---|
| FF00 | 4C00E0 | JMP RESET | Drive Kaltstart | | |
| FF03 | 4C73E0 | JMP RESET2 | Warmstart | | |
| FF06 | 4C8CE0 | JMP BEREIT | Bereitschaftroutine | | |
| FF09 | 4C41E1 | JMP MOTON | Motor zwingend einschalten | A,Y | |
| FF0C | 4C3CE1 | JMP TSTMON | Motor einschalten wenn Klappe zu ist | A,Y | Z=0 |
| FF0F | 4C2DE1 | JMP MOTOFF | Motor ausschalten | X,Y | |
| FF12 | 4CD2E3 | JMP SDELAY | Motor Timer stellen | X,Y | |
| FF15 | 4CA9E1 | JMP SDRDDP | Drive Density stellen und anzeigen | | |
| FF18 | 4CF2E2 | JMP XWAIT | Warteschleife kurz | Y | |
| FF1B | 4CF0E2 | JMP X2WAIT | Warteschleife lang | Y | |
| FF1E | 4CD1E2 | JMP TRACK0 | Track 0 positionieren | | |
| FF21 | 4C2FE3 | JMP TRADJA | Track # anzeigen und Kopf positionieren | | N=1 |
| FF24 | 4C32E3 | JMP TRADJ | Kopf positionieren, Track # nur anzeigen, wenn Trackwechsel | | N=1 |
| FF27 | 4C0FE3 | JMP TRVR | 1 Step vorwaerts oder rueckwaerts gehen | X,Y | |
| FF2A | 4C7BE3 | JMP CONRES | Disk Controller stoppen | X,Y | |
| FF2D | 4C78E3 | JMP CONRE2 | 2 mal CONRES | X,Y | |
| FF30 | 4C85E3 | JMP WREADY | auf Controller 'In Use'-Flag=0 warten | X,Y | |
| FF33 | 4CB3E4 | JMP RD128B | 128 Bytes vom Computer nach EXBUF holen | | Bit 6 CONST ($11) |
| FF36 | 4CB6E4 | JMP RD256B | 256 Bytes vom Computer nach EXBUF holen | | Bit 6 CONST ($11) |
| FF39 | 4CBCE4 | JMP RDBTS | Accu=Anzahl der Bytes nach Buffer (X/Y-Register) holen | | Bit 6 CONST ($11) |
| FF3C | 4CCDE1 | JMP RDSFOL | nach Verzoegerung Sectorfolge vom aktuellen Track lesen | Y | C=1 |
| FF3F | 4CD2E1 | JMP RDSFO1 | sofort Sectorfolge vom aktuellen Track lesen | Y | C=1 |
| FF42 | 4C6FE2 | JMP RDTRA | alle Sectoren des aktuellen Track ins RAM einlesen | | C=1 |
| FF45 | 4C44E2 | JMP RDTRAV | wie RDTRA aber mit Verify und einem Retry | | C=1 |
| FF48 | 4C64EB | JMP TSTWR | noch zu schreibende Sectoren aus RAM auf Diskette schreiben | | Z=0 |
| FF4B | 4C04E6 | JMP TSTDAT | TSTWR ausfuehren und alle Sectoren als nicht gelesen markieren | | Z=0 |
| FF4E | 4C12E5 | JMP SD128B | 128 Bytes vom EXBUF zum Computer senden | | |
| FF51 | 4C15E5 | JMP SD256B | 256 Bytes vom EXBUF zum Computer senden | | |
| FF54 | 4C1BE5 | JMP SDBTS | Accu=Anzahl der Bytes aus Buffer (X/Y-Reg.) senden | | |
| FF57 | 4CF0E4 | JMP SEND41 | 'A' zum Computer senden | | |
| FF5A | 4CFFE4 | JMP SEND43 | 'C' zum Computer senden | | |
| FF5D | 4C03E5 | JMP SEND45 | 'E' zum Computer senden | | |
| FF60 | 4C07E5 | JMP SEND4E | 'N' zum Computer senden | | |
| FF63 | 4CCFEA | JMP RDSECT | aktuellen Sector von Diskette in vorbezeichneten RAM einlesen | X | Z=0 |
| FF66 | 4CD4EA | JMP RDSEC1 | bezeichneten Sector in bezeichneten RAM einlesen | X | Z=0 |
| FF69 | 4C69EC | JMP WRSECT | aktuellen Sector von vorbezeichneter RAM-Adr. auf Disk schreiben | X | Z=0 |
| FF6C | 4C6EEC | JMP WRSEC1 | bezeichneten Sector von vorbezeichneter RAM-Adr. schreiben | X | Z=0 |
| FF6F | 4C88E4 | JMP TSTWRP | Write Protect und Klappe testen | X,Y | Z=0 |
| FF72 | 4CA4EC | JMP VERSEC | aktuellen Sector mit angegebenem RAM vergleichen | X | Z=0,C=1 |
| FF75 | 4CA9EC | JMP VERSE1 | bezeichenten Sector mit angegebenem RAM vergleichen | X | Z=0,C=1 |
| FF78 | 4C4CE5 | JMP STELL | COM-Status auf 'Error' und 2 Retry's setzen | X,Y | |
| FF7B | 4C55E5 | JMP QUITT | Quittung 'C' oder 'E' je nach COM-Status senden | Y | |
| FF7E | 4C3DE8 | JMP RDHEAD | Die nachsten 'Header'-Daten lesen | Y | C=1,Z=0 |
| FF81 | 4C42E8 | JMP RDHD1 | wie RDHEAD aber Timer nicht setzen | Y | C=1,Z=0 |
| FF84 | 4C1DE8 | JMP RDHDSP | Kopf positionieren und nachsten 'Header' lesen | | C=1 |
| FF87 | 4CE4E9 | JMP CALCTS | Track- und Sectornummer errechnen | X | C=1,N=1,Z=0 |
| FF8A | 4C1AEA | JMP SETBUF | Buffer nach aktuellem Sector setzen | X,Y | |
| FF8D | 4C1CEA | JMP SETBUF2 | Buffer nach Sectornummer im Accu setzen | X,Y | |
| FF90 | 4C08E9 | JMP SEXBUF | Adresse des Extended-Buffers setzen | X,Y | |
| FF93 | 4C2CEA | JMP SETRWL | Anzahl der Bytes fuer zu uebertragenden Datenblock setzen | X | |
| FF96 | 4C6BE0 | JMP COPSLT | Sectorliste fuer aktuelles Density in Zeropage kopieren | | |
| FF99 | 4C82EF | JMP BELL1 | 1 Bell (Buzzer) ausgeben | Y | |
| FF9C | 4CC7EF | JMP CLRDSP | Display abschalten | A,X,Y | |
| FF9F | 4CF6EF | JMP TRAANZ | aktuelle Track # anzeigen | A | |
| FFA2 | 4C07F0 | JMP DEZOUT | Wert im Accu in dezimaler Form anzeigen | | |
| FFA5 | 4CDFEF | JMP HEXOUT | Wert im Accu in Hexadezimaler Form anzeigen | Y | |
| FFA8 | 4C1FF0 | JMP DENDSP | aktuelles Density anzeigen | Y | |
| FFAB | 4C72E5 | JMP SETTIM | Timer mit Wert im Accu setzen | A,X,Y | |
| FFAE | 4C0FEF | JMP CLRTRA | Einen Track mit unlesbarem Format versehen (reformatieren) | | C=1 |
| FFB1 | 4C01EF | JMP CLRDSK | ganze Diskette reformatieren | | C=1 |
| FFB4 | 4CC6F2 | JMP RAMTST | Einsprung fuer RAM-Test, 2 Bytes werden gesendet | | Bit 7 CONST ($11) |
| FFB7 | 4CA1F2 | JMP ROMTST | ROM-Test Einsprung, es wird nur quittiert | | Bit 7 CONST ($11) |
| FFBA | 4C22F3 | JMP SPEEDT | Einsprung fuer Speed-Test, 2 Bytes werden gesendet | | Bit 6+7 CONST ($11) |